An Introduction to the ANSI N42.42 Data File Format

George Lasche

Bob Huckins

Course Outline

- Introduction
- Basic XML concepts
- ANSI N42.42 format elements
- Walkthrough of sample N42 files
- Example software for writing and reading N42 files

Your Instructors

George Lasche

Physicist

Sandia National Laboratory

Bob Huckins

Engineer

Canberra Industries

The N42.42 Working Group

- George Lasche, SNL, *Co-chair and project leader*
- Leticia Pibida, NIST, *Co-chair and project leader*
- Bob Huckins, Canberra, *Secretary and editor*
- Robert Bass, PNNL
- Peter Chiaro, ORNL
- Carl Czajkowski, BNL
- Charles Finfrock, BNL
- Ronald Keyser, Ortec
- Johnny Long, Thermo
- Lew Meixler, NucSafe
- Keith Olson, LANL
- Scott Rogers, Canberra
- Richard Smola, Ludlum
- Adrian Stoian, Exploranium/SAIC
- Dave Weirup, LLNL

What is N42.42?

- **N42.42 accommodates all 5 of the basic Homeland Security instrument types:**
  - **N42.32: Alarming Personal Radiation Detectors**
  - **N42.33: Portable Radiation Detection Instrumentation**
  - **N42.34: Radionuclide Identifier Detectors**
  - **N42.35: Radiation Detection Portal Monitors**
  - **N42.38: Spectroscopy-Based Portal Monitors**

The Goals of ANSI N42.42

- **"Provide a standard data interchange format for DHS radiation instruments": i.e., a standard file format**
- **Vendor-neutral**
- **Text based – readable without special software**
- **Extensible**
- **"Validatable" (to some extent, at least)**

These goals pointed to the use of XML

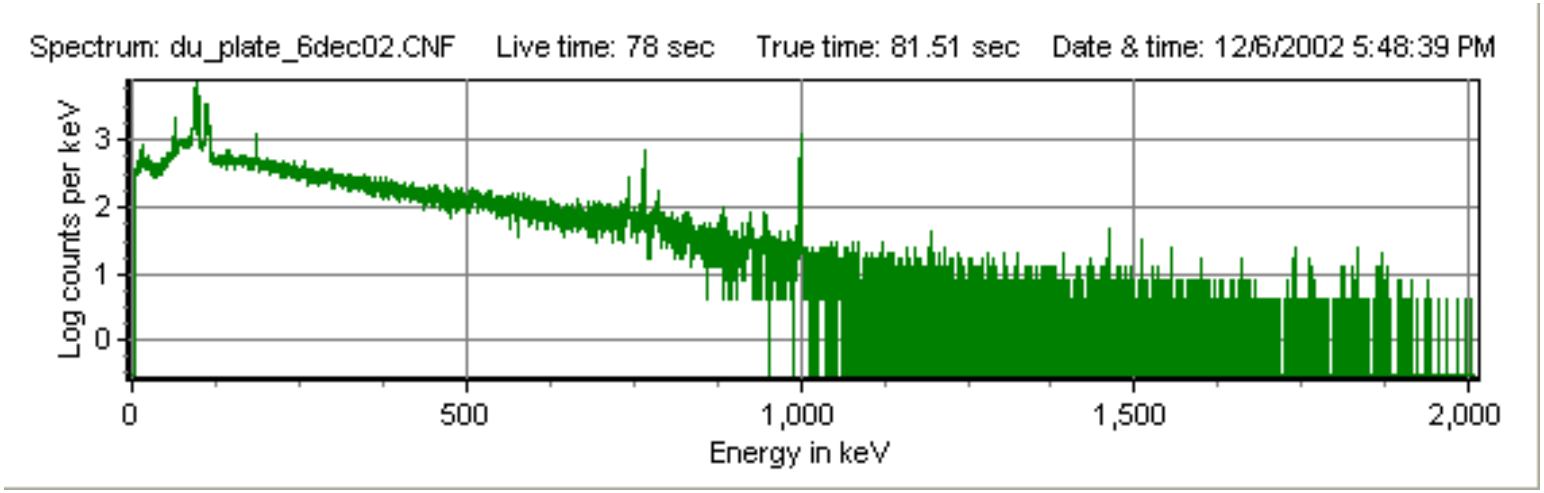
The "Non-Goals" of ANSI N42.42

- **The definition of communications with instruments (i.e., TCP/IP, UPnP, etc.)**
- **A rigid definition of how the instruments should operate**
- **Incorporation of data specific to a unique implementation or algorithm**

N42 Format Key Features

- **Extensible to unforeseen future needs – retains backward *and forward* compatibility**
- **Human readable – in an emergency, no special software is needed to get the key data**
- **Machine readable – based on XML, a standard maintained by the World-Wide Web Consortium (W3C)**
- **Provides a single format for all radiation detectors – all data can be archived in one common format – XML acts also as a database, with retrieval using XPath**
- **Not binary -- Easily passes servers scanning for viruses**
- **File size economy – Generally smaller files than the binary equivalent**
- **Can be checked (”validated”) by machine for correct syntax**

N42.42 Files can be smaller than binary counterparts



- Canberra CNF format: 63 kB
- Ortec CHN format: 33 kB
- N42 format: 22 kB

A Brief History of XML

- 1960s: IBM creates Generalized Markup Language (GML)
  - Separates document content from formatting
- 1980s: GML → Standardized GML (SGML), ISO 8879:1986
  - Facilitates sharing of machine-readable information: a standard way of describing information
- Early 1990s: Hypertext Markup Language (HTML) is loosely based on SGML
  - Created for Internet information distribution
- Late 1990's: eXtensible Markup Language (XML) is based on SGML, compatible with HTML
  - More lightweight than full SGML

XML Basics
The XML Format

- XML, like HTML, is text-based and is machine and human readable
- Everything in an XML file is contained in an “element”:

  <name attribute="value">content</lname>

- Elements (usually) begin and end with “start” and “end” tags in angle brackets
- Elements have names, and may have values and attributes.
- A simple example:

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
     <title>Basic bread</title>
     <ingredient amount="3" unit="cups">Flour</ingredient>
     <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
     <ingredient amount="1.5" unit="cups" state="warm">Water</ingredient>
     <ingredient amount="1" unit="teaspoon">Salt</ingredient>
     <instructions>
          <step>Mix all ingredients together, and knead thoroughly.</step>
          <step>Cover with a cloth, and leave for one hour in warm room.</step>
          <step>Knead again, place in a tin, and then bake in the oven.</step>
     </instructions>
</recipe>
```

Start Tag → `<title>` … `</title>` ← *End Tag*

XML Schema

- The syntax of XML data can be defined using “XML Schema”
- XML parsers can use a schema file to validate the syntax of a given XML file
- A schema file has been created for the ANSI N42.42 format and is available on the NIST web site
- Custom schema files can be created in order to extend the format

XML Datatypes

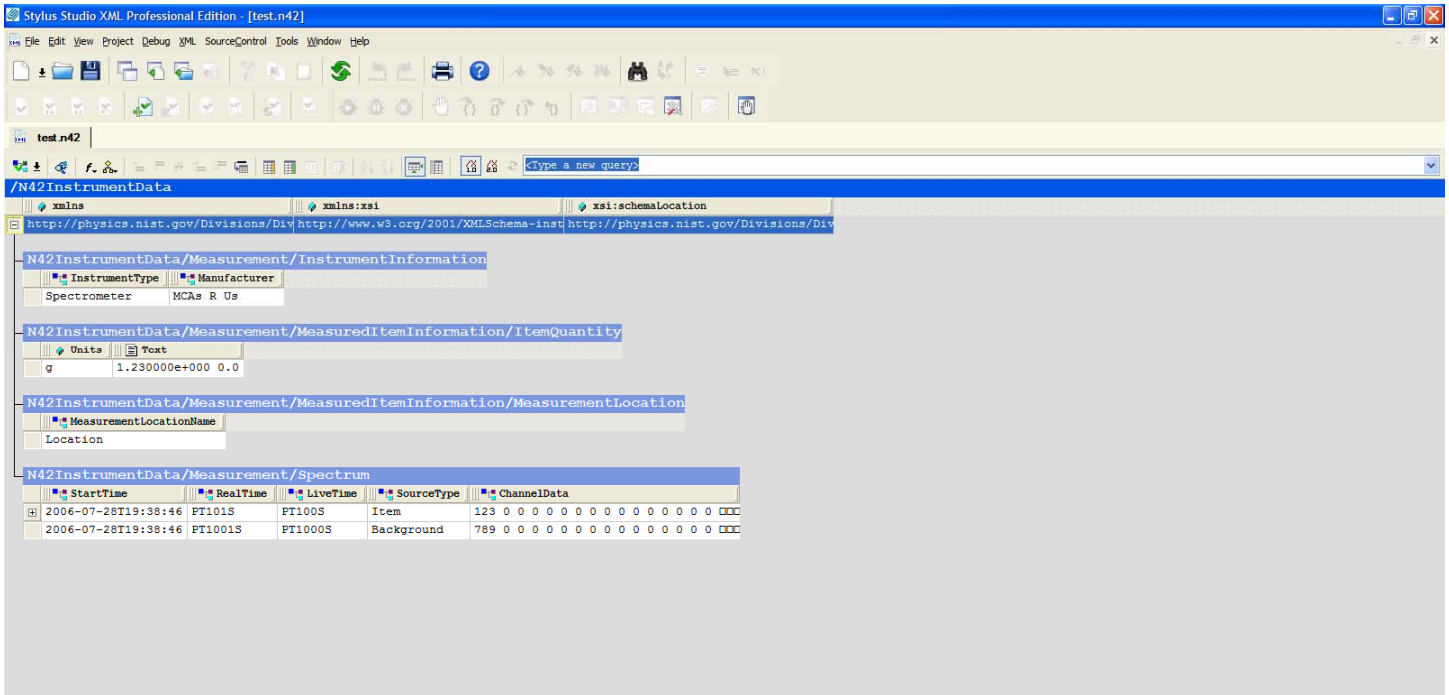
- Even though XML is all text, XML Schema defines “datatypes”
- Examples of datatype *double*:
  - 2
  - 2.23
  - .223e+01

  The number of digits should reflect the precision with which the value is known.
- Examples of datatype *positiveInteger*:
  - 2
  - 1001
- Example of datatype *datetime*: 2006-07-28T19:38:46
- Example of datatype *duration*: PT99.5S
- Datatype *string* can be anything!
- XML Schema allows custom datatypes to be defined; this has been done for N42 – for example, *doubleUnc* is a value plus its uncertainty: 1.234 0.123

Other XML Features

- **Authentication: XML Signature allows all or just parts of an XML file to be digitally signed**
- **Encryption: XML Encryption allows all or just parts of an XML file to be encrypted**

- **Editors are available that "understand" XML:**
  - **Display file contents graphically (see below)**
  - **Autocomplete elements**
  - **Check for correct general XML syntax**
  - **Check for correct specific XML syntax per a schema file**



How to Use the N42 Standard Document

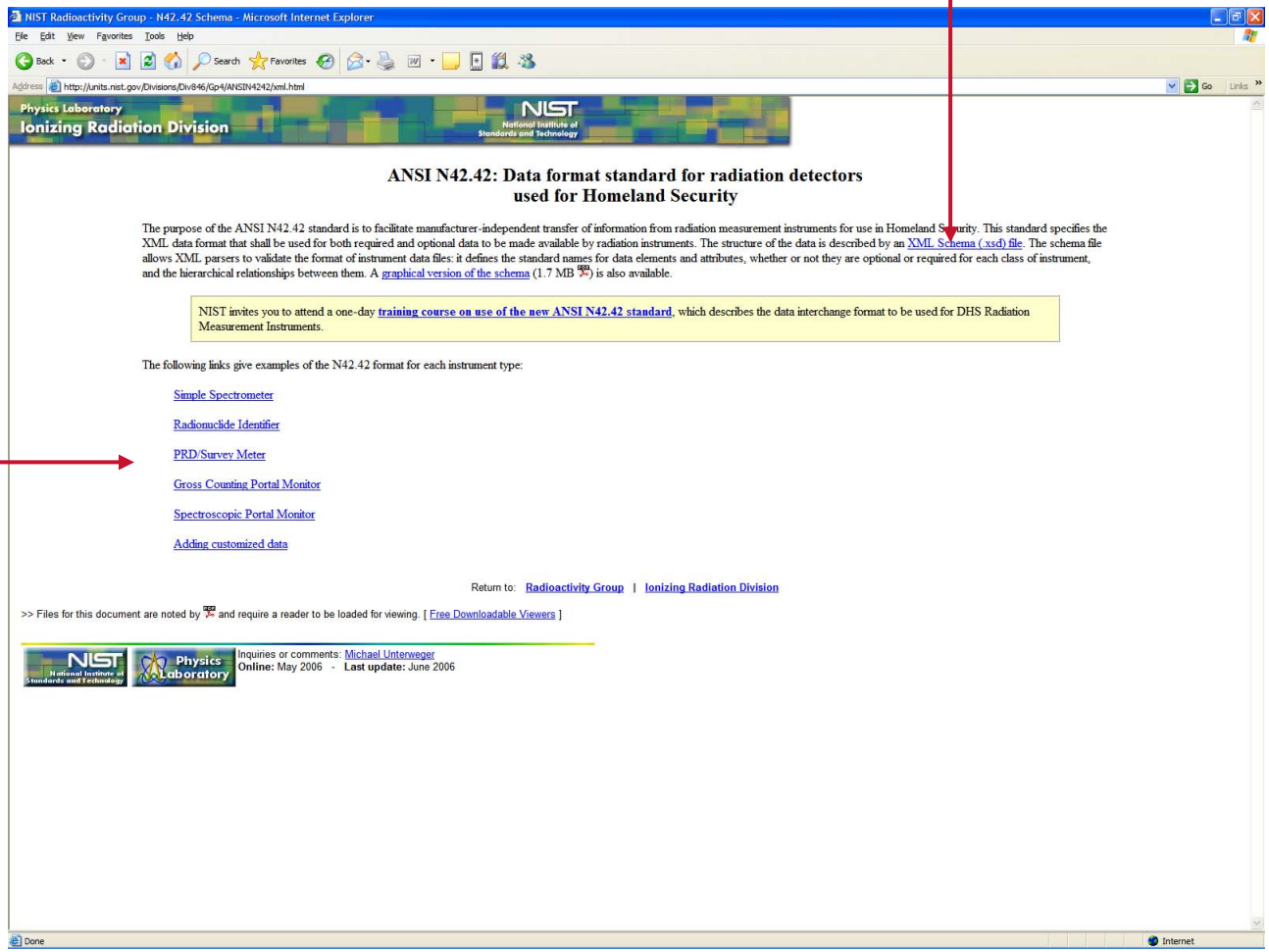
- Look at the examples in Annexes B-F first
- Each element is described in the body of the standard:
  - It's parent element(s)
  - Its child elements
  - Data type of its contents, if any allowed
  - What attributes it accepts
  - How many times it can appear:
    - Zero or once (is optional, can appear only once)
    - Zero or more (is optional, can appear multiple times)
    - Once (i.e., it is required, can appear only once)
    - One or more (i.e., it is required, can appear multiple times)
- Note that elements must appear in the order that they are listed in *Child Elements* description

ANSI N42.42 on the Web

http://units.nist.gov/Divisions/Div846/Gp4/ANSIN4242/xml.html

Link to schema file

Links to examples



Walk through a Simple Spectrum File: Annex B

- Annex B of the standard is an example 256 channel spectrum with minimal “optional” information
- Bob will go through the file line-by-line
- George will display the relevant parts of standard
- We’ll repeat the process for:
  - Survey Meter/PRD (Annex F)
  - Gross Counting Portal Monitor (Annex D)
  - Spectroscopic Portal Monitor (Annex E)

Extending the Format: Adding data that is not in the standard

- **OK, OK, we didn't include *every* possible kind of data in the standard**
- **New elements and attributes can be added using "custom namespaces"**
- **They can also be validated with a custom schema**
- **Annex G is an example of adding a custom attribute – we'll walk through it now**
- **The DNDO ASP ICD also defines additional information**

How you can contribute to the future of the standard

- **Contact with suggestions:**
  - **George Lasche (Sandia): gplash@sandia.gov**
  - **Leticia Pibida (NIST): leticia.pibida@nist.gov**
  - **Bob Huckins (Canberra): bhuckins@canberra.com**
- **Join the N42.42 Working Group!**

Using Cambio

- Cambio is an application intended to automatically read and display any spectrum file of any format in the world that the nuclear emergency response community might encounter – including unknown types.
- Cambio has a batch processing capability to automatically translate a large number of archival spectral files of any format to the N42.42 format.
- Cambio is unrestricted unclassified and distributed by FTP downloads with email notifications approximately every 6 weeks as new formats are identified and new capabilities are provided.

Cambio Supported File Formats as of 6 Feb 2006

- ANSI N42.42
- Aspect MKC (A-02, A-03)
- Berkeley Nucleonics SAM-935 with QCC decompression and linearization
- Berthold LB-125 (includes multi-record files)
- Canberra CAM format (*.cnf) (includes detection of “counterfeit” variants)
- Canberra Accuspec (*.dat)
- Canberra Programmers Toolkit (*.tka)
- Canberra Inspector 1000 specific CAM format (*.cnf)
- CTC “MCS”
- Davidson (4 variants)
- Exploranium (multi-record files): Gr-130, Gr-135 v1 (2 variants), Gr-135 v2 (2 variants), ASCII (3 variants)
- FieldSpec (now IdentiFinder): Native SPC, FieldSpec-N variant, SPE
- GADRAS (includes multi-record files): PCF, PCC (RAID, RIS, SMART), ASC
- IAEA Generic SPE; Mini-MCA (*.spe); 3 methods date and time; 3 polynomial energy calibration; energy calibration from a series of channel-energy pairs
- LANL: GN-2 (multirecord); GN-3 (4 variants) (multi-record); Palm Pilot
- Ortec: CHN, SPC (both integer and floating), SPE, Print-to-file ASCII
- PDR-78 (file extensions are actually sequence numbers)
- PGT Avalon (*.ans): Revision 1; Revision 2 and above
- Quantrad Ranger (multi-record)
- Rainbow Model 7010
- RobFit (FREE, HDTA, Z4DA; real and integer)
- SAIC RadSmart
- STE Pager-X
- STL Cadillac (ASC and CSV)
- STL Yugo
- Target NanoSpec
- XIA Polaris (*.itx)
- XRF ICS-4000 (2 variants)

How to get on the Cambio distribution list

- Send an email to George Lasche at gplasch@sandia.gov
- No restrictions
- No fees
- No questions asked

Writing N42 Format Files

- The simplest way (conceptually): ad-hoc – just grunt out code to do it!
  - Easy to understand
  - No new tools/techniques required
- Use a tool like DOM:
  - Makes it easy to read in and modify a file

Writing N42: A Simple C Example

- BasicWriteSpectrumApp is a MS Visual Studio project that includes routines to write N42 spectrum files
- This code is a gift to the community and is freely distributed: it can be used/cloned/modified without restriction or attribution
- Code is straight C, nothing fancy…
- The data to be written is put in a structure that is passed to a subroutine, which blasts it out using standard C RTL stream I/O

Reading N42 Files

- Ad-hoc
- Use a standard tool or API:
  - DOM: reads in entire file and contructs a “node tree” that can be traversed
  - SAX: reads in the file one element at a time and generates events

  DOM is easy to understand and use but will have problems with really big files. SAX can handle files of infinite size but is harder to use.

Reading N42: Using the DOM API

- **DOMExample includes:**
  - **The Apache Organization Xerces XML parser distribution (supports both DOM and SAX APIs)**
  - **ReadSpectrum: a MS Visual Studio project that uses the Xerces DOM to read an N42 file and print out the contents of the spectrum it contains (if any)**
  - **Xerces is open-source and can be used under the Apache license**
  - **ReadSpectrum is a gift to the community: it can be used/cloned/modified without restriction or attribution**

Thank You!

- Thanks for coming!
- How can we make this course better?